Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method for protecting data as it passes between ~~through a buffering device that connects~~ first and second protocols that use different block sizes or unblocked data, said method comprising:

connecting a protocol interface device between a first device that uses the first protocol and a second device that uses the second protocol, and connecting a memory device to the protocol interface device;

said protocol interface device including a cyclical redundancy code (CRC) engine;

allocating a plurality of blocks of storage in a first memory in the memory device ~~said buffering device~~ for storing a transfer length of unblocked data;

receiving said data on a first port of said protocol interface device from the first device ~~from a source~~, said first device ~~source~~ determining said transfer length;

said protocol interface device writing said data, as said data is received, to successive ones of said plurality of blocks until an end of said transfer length is reached , each one of said plurality of blocks having a length of 2^n, where n is a positive integer;

if the CRC engine determines that an end one of said plurality of blocks that includes an end of said transfer length is not full of data, adding padding to said end one of said plurality of blocks until said end of said plurality of blocks is complete, wherein padding is added only to an end one of said plurality of blocks that includes an end of said transfer length until the end one of the plurality of blocks is complete;

as said data is being written to each one of said plurality of blocks;[[,]] calculating, by said CRC engine, a running cyclical redundancy code for each of said plurality of blocks by calculating an intermediate cyclical redundancy code after each byte of said data is written to each one of the plurality of blocks;

after each intermediate cyclical redundancy code is calculated: storing, in a CRC memory that is included in the protocol interface device, said intermediate cyclical redundancy code along with information that identifies said first port, the CRC memory being separate from but connected to said CRC engine;

when writing to each one of said plurality of blocks is completed, storing, for each one of said plurality of blocks, a final value of said running cyclical redundancy code that was calculated for each one

of said plurality of blocks as a first cyclical redundancy code in a second memory in said memory device ~~on said buffering device~~; and

when ~~writing~~ said data is to be written to a second port of the protocol interface device: using said CRC engine to compute ~~computing~~ a second cyclical redundancy code for each of said plurality of blocks; comparing, using a comparator that is included in the protocol interface device and coupled to the CRC engine, ~~and if~~ said second cyclical redundancy code corresponding to a given block ~~is equal~~ to said first cyclical redundancy code corresponding to said given block; and if said second cyclical redundancy code corresponding to the given block is equal to said first cyclical redundancy code corresponding to said given block, writing said given block to said second port.

2. (Original) The method of Claim 1, wherein the data is received with a protection code that is checked and discarded.

3. (Currently amended) The method of Claim 1, wherein said memory ~~buffering~~ device is a DDR device, the first device is ~~connected between~~ a bus, and the second device is [[and]] a tape drive.

4. (Currently amended) The method of Claim 1, further comprising:

coupling said first device to said first port, and coupling said second device to said second port;

the protocol interface device including a selector switch that is coupled to said first port and said second port;

coupling said selector switch to said CRC engine;

using the selector switch to select one of the first port and the second port to which to couple the CRC engine, said CRC engine coupled to only one of the first and second ports at a time; and

the CRC engine communicating with said first device and said second device only through said selector switch.

~~wherein one of said first and said second ports is connected to a protocol that does not use fixed block lengths.~~

5. (Original) The method of Claim 1, wherein locations in said second memory are mapped to locations in said first memory.

6. (Canceled)

7. (Currently amended) A device for buffering data between two protocols, at least one of which does not utilize blocks, said device comprising:

a first port connected to communicate using a first protocol of said two protocols;

a second port connected to communicate using a second protocol of said two protocols;

a cyclical redundancy code (CRC) engine connected to be selectively connected to one of said first port and said second port;

a plurality of blocks of storage that have been allocated for storing a transfer length of unblocked data;

a first random access memory connected to said cyclical redundancy code engine and including said plurality of blocks for storing said data that is passing between said first port and said second port, said data being written to successive ones of said plurality of blocks as said data is received until an end one of said transfer length is reached, each one of said plurality of blocks having a fixed size;

if said CRC engine determines that an end one of said plurality of blocks that includes an end of said transfer length is not full of data, padding being added to said end one of said plurality of blocks until said end of said plurality of blocks is complete, wherein padding is added only to an end one of said plurality of blocks that includes an end of said transfer length;

said cyclical redundancy code engine calculating a running cyclical redundancy code for each one of said plurality of blocks as data is written to each one of said plurality of blocks by calculating an intermediate cyclical redundancy code after each byte of said data is written to each one of the plurality of blocks;

a CRC memory for, after each intermediate cyclical redundancy code is calculated, storing said intermediate cyclical redundancy code along with information that identifies one of the first or second ports through which said data was received;

a second random access memory connected to said cyclical redundancy code engine for storing, for each one of said plurality of blocks, a final value of said running cyclical redundancy code that was calculated for each one of said plurality of blocks as a first cyclical redundancy code when writing to each one of said plurality of blocks is completed; and

a comparator connected to compare a second cyclical redundancy code calculated for each one of said plurality of blocks with said first cyclical redundancy code calculated when writing to each one of said plurality of blocks is completed written;

whereby the data passed through said device is protected by a cyclical redundancy code.

8. (Original) The device of Claim 7, wherein said cyclical redundancy codes are stored in said second random access memory in a mapped relationship to said fixed size blocks stored in said first random access memory.

9. (Original) The device of Claim 7, further comprising a protection module connected to said first port for checking a protection code that is received and discarding said protection code.

10. (Original) The device of Claim 7, wherein locations in said second random access memory are mapped to locations in said first random access memory.

11. (Canceled)

12. (Currently amended) A computer program product on a computer-readable device, comprising the computer implemented steps of:

a protocol interface device connected between a first device that uses the first protocol and a second device that uses the second protocol, and connecting a memory device to the protocol interface device;

said protocol interface device including a cyclical redundancy code (CRC) engine;

first instructions for allocating a plurality of blocks of storage in a first memory in the memory device for storing a transfer length of unblocked data;

second instructions for receiving said data on a first port of said protocol interface device from the first device ~~from a source~~, said first device ~~source~~ determining said transfer length;

third instructions for writing, by the protocol interface device, said data to a first memory in a buffering device, as said data is received, to successive ones of said plurality of blocks until an end of said transfer length is reached, each one of said plurality of blocks having a length of 2^n, where n is a positive integer;

if the CRC engine determines that an end one of said plurality of blocks that includes an end of said transfer length is not full of data, fourth instructions for adding padding to said end one of said plurality of blocks until said end one of said plurality of blocks is complete, wherein padding is added only to an end one of said plurality of blocks that includes an end of said transfer length;

fifth instructions for calculating, by the CRC engine, a running first cyclical redundancy code for each of said plurality of blocks as data is being written to each one of said plurality of blocks by calculating an intermediate cyclical redundancy code after each byte of said data is written to each one of the plurality of blocks;

after each intermediate cyclical redundancy code is calculated: sixth instructions for storing, in a CRC memory that is included in the protocol interface device, said intermediate cyclical redundancy code along with information that identifies said first port, the CRC memory being separate from but connected to said CRC engine;

seventh [[sixth]] instructions for when writing to each one of said plurality of blocks is completed, storing, for each one of said plurality of blocks, a final value of said running cyclical redundancy code that was calculated for each one of said plurality of blocks as a first cyclical redundancy code in a second memory in said memory device ~~on said buffering device~~; and

when said data is written to a second port of the protocol interface device: eighth instructions for using said CRC engine to computer ~~seventh instructions for computing, when writing said data to a second port,~~ a second cyclical redundancy code for each of said blocks of said plurality of blocks; ninth instructions for comparing, using a comparator that is included in the protocol interface device and coupled to the CRC engine, ~~and if~~ said second cyclical redundancy code corresponding to a given block ~~is equal~~ to said first cyclical redundancy code corresponding to said given block; and if said second cyclical redundancy code corresponding to the given block is equal to said first cyclical redundancy code corresponding to said given block, tenth writing said given block to said second port.

13. (Previously presented) The computer program product of Claim 12, wherein said second instructions check a protection code received with the data and discarded said protection code.

14. (Currently amended) The computer program product of Claim 12, wherein ~~said computer program product is embodied on a protocol interface device connected between~~ said first device is a bus and the second device is a tape drive

15. (New) The method according to claim 4, further comprising:

said selector switch being used to select the second port;

in response to said second port being selected, receiving new data on the second port from said second device;

said protocol interface device writing said new data, as said data is received, to successive ones of a second plurality of blocks until an end of said transfer length is reached, each one of said second plurality of blocks having a length of 2^n, where n is a positive integer;

as said data is being written to each one of said second plurality of blocks: calculating, by said CRC engine, a second running cyclical redundancy code for each of said second plurality of blocks by

calculating a second intermediate cyclical redundancy code after each byte of said data is written to each one of the plurality of blocks; and

after each second intermediate cyclical redundancy code is calculated: storing, in a CRC memory that is included in the protocol interface device, said second intermediate cyclical redundancy code along with information that identifies said second port.

16. (New) The device according to claim 7, further comprising:

a first device coupled to said first port;

a second device coupled to said second port;

a selector switch coupled to said first port, said second port, and said CRC engine;

the selector switch for selecting one of the first port and the second port;

said CRC engine coupled to only one of the first and second ports at a time; and

the CRC engine communicating with said first device and said second device only through said selector switch.